NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMPLEMENTING METOC TRANSFORMATION: APPLYING AUTONOMOUS AGENTS

by

J. Vorrath

September 2004

Thesis Advisor:	Carlyle Wash
Second Reader:	Neil Rowe

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2004	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Implementing METOC Transformation: Applying Autonomous Agents			**5. FUNDING NUMBERS**
6. AUTHOR(S) J. Vorrath, LT, USN			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

To achieve integration of the Naval Meteorology and Oceanography (METOC) community into the developing FORCEnet environment, transformational innovations must be researched and implemented. Agent based software is an example of technology that can be employed in this way by changing the method by which METOC data is distributed to end-users. This thesis documents the creation and implementation of a software agent that uses Internet connections to retrieve numerical model data, loads this output into array data containers, and then makes it available to the end-user in a machine-readable forecast object format. The impact of the importation of this forecast object into warfare commander command-and-control software is then assessed using the commercially available SEAWAY logistics tool. This assessment highlights the importance of defining the METOC functional requirements for the emerging FORCEnet environment, so that proper interfaces to exchange data freely, and visually depict it, are incorporated during next generation software development. Using these types of agents to automate the generation and delivery of weather parameters could also allow the importation of data into previously insular software, provide reach-back support to the warfighter, and be a means of reducing manpower and budgetary requirements during this time of fiscal constraint.

14. SUBJECT TERMS METOC, Transformation, FORCEnet, Software Agents, SEAWAY.			**15. NUMBER OF PAGES** 65
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release, distribution is unlimited

IMPLEMENTING METOC TRANSFORMATION: APPLYING AUTONOMOUS AGENTS

Jonathan J. Vorrath
Lieutenant, United States Navy
B.S., United States Naval Academy, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN METEOROLOGY AND PHYSICAL OCEANOGRAPHY

from the

NAVAL POSTGRADUATE SCHOOL
September 2004

Author: J. Vorrath

Approved by: Dr. Carlyle H. Wash
Thesis Advisor

Dr. Neil C. Rowe
Second Reader

Dr. Philip A. Durkee
Chairman, Department of Meteorology

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

To achieve integration of the Naval Meteorology and Oceanography (METOC) community into the developing FORCEnet environment, transformational innovations must be researched and implemented. Agent based software is an example of technology that can be employed in this way by changing the method by which METOC data is distributed to end-users. This thesis documents the creation and implementation of a software agent that uses Internet connections to retrieve numerical model data, loads this output into array data containers, and then makes it available to the end-user in a machine-readable forecast object format. The impact of the importation of this forecast object into warfare commander command-and-control software is then assessed using the commercially available SEAWAY logistics tool. This assessment highlights the importance of defining the METOC functional requirements for the emerging FORCEnet environment, so that proper interfaces to exchange data freely, and visually depict it, are incorporated during next generation software development. Using these types of agents to automate the generation and delivery of weather parameters could also allow the importation of data into previously insular software, provide reach-back support to the warfighter, and be a means of reducing manpower and budgetary requirements during this time of fiscal constraint.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Dr. Carlyle Wash. Without his willingness to invest his valuable time and considerable talents in a thesis idea that was well outside the established norm, none of this would have been possible. As problems were encountered, his deft guidance was invaluable in keeping the completion of this project on track. Thanks to another one of his thesis students as well, Vic Ross. He blazed the way for the METOC/CS students who followed behind him.

Priceless support for the students attempting to complete this unusual set of degrees also came from CDR Joseph. Without his intervention on our behalf, our efforts most likely would have cut short. He was able to see beyond the short-lived difficulties associated with class scheduling and officer detailing, and see how an 1800 with this sort of talent-base will ultimately benefit our community.

This project would not have gotten underway without the support of CDM Technologies, Inc. of San Luis Obispo. Dr. Jens Pohl and Kym Pohl graciously donated some of their valuable time to discuss agent and ontology technologies. The talents of Lara Fields were absolutely indispensable. She spent several hours demonstrating SEAWAY and assisting me in defining the structure of my proposed agent.

This thesis would not have been possible without the assistance of several members of the Fleet Numerical and Meteorology Center. CAPT Gunderson introduced me to the idea of transformation, and helped me understand how it might possibly be accomplished. Mr. Clancy and Darin Keeter took the time on several occasions to discuss data service concepts. Doug Gentges donated much of his original Java code to this project, and then patiently answered all of my many and varied questions about it.

Of course, I would especially like to thank my family, friends, and loved ones. Without their steadfast support, loving encouragement, and occasional boots to the rear, I would not have been able to complete this project.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Within the context of our advancing information age, modern warfare commanders must manage and interpret data effectively or they may become overwhelmed by it. In many cases a warfare planner will have an overwhelming abundance of data available concerning his target. As a result, it becomes the task of his staff, or supporting commands, to sift through the available data, produce meaningful information, and "bubble to the surface" the vital 10% of that information that the commander must be aware of in order to make able decisions concerning the tactical employment of assets. In his class on Autonomous Agents at the Naval Postgraduate School, Prof. John Hiles referred to this as an attention focus system. This view of an attention focus system that feeds information to a warfare planner can be extended to include meteorological data as well. Within this discipline, huge stores of database numeric model output are already available and better means of automating the interpretation of these is required.

In addition to receiving focused, relevant information, warfare commanders must be given that information in the format that is most applicable to them. In some cases this may be in the form of verbal briefings from an attached Meteorology and Oceanography (METOC) Officer as the operational and meteorological picture develops. In many other cases this may mean providing utility so that commanders can display environmental conditions and their impact upon assets within the same software they utilize for planning. Ultimately how that information is provided should depend only on the preferences of the customer, not on the limitations of our system of dissemination. Indeed, as the sciences that define how we handle, process, and pass information continue to advance at an astounding rate, commanders should have access to whatever information they need in whatever manner they desire.

As stated above, then, there are two central issues to be examined when considering how best to deal with meteorological data. The first is how that data is processed and focused into relevant information. The second is how to pass that

information on to warfare commanders that makes use of a flexible architecture that takes advantage of our progressing information age.

B. PROCESSING DATA AND PASSING INFORMATION – THE WAY IT IS

In order to develop and validate new ways of processing and passing meteorological information, we must at least briefly examine the manner in which we currently perform these tasks. In the U.S. Navy, meteorology professionals currently receive numeric model output, review it, and then provide their customers with forecast information in one of several ways. The most basic example of a meteorology product is a forecast of environmental conditions for a given location and time. This is a forecast that may be for a specific point or may cover a larger region, but in either case consists of a simple listing of parameters such as temperature, wind speed, etc. A more evolved type of operational forecast is one that considers the impact of meteorological conditions on a specific type of asset or platform that is designed for use in a certain manner during a planned operation. This may take the form of a "stoplight" type graphic, which we will discuss further shortly, or be represented as a tactical decision aid (TDA) of some type. A third method of providing forecast information, and the one most often preferred by customers, is to have a forecaster on-scene working directly with the operators during the development and execution of a military exercise or mission. Each of these methods has advantages and disadvantages, which will be examined individually before an alternate means of producing and disseminating meteorological forecasts is proposed.

It is important to first consider how meteorological data is currently processed within the Navy METOC community. Today, the officer and enlisted personnel that comprise the METOC community wear many hats. Along with a very wide range of tasking, however, their main duty has always been to produce forecasts of environmental conditions and often to also evaluate how these factors will impact a mission. In a sense, they take the raw numeric model output, apply to this data a nuanced understanding of both local effects and model tendencies, and then produce usable information in terms of an environmental forecast or impact assessment. The important point to emphasize here is that the data is essentially processed, given context, and evaluated within that human forecaster's mind.

As we examine the forms in which forecasts are sent to customers within the Navy it is logical to begin with one of the most widely used and traditional methods. This means of forecasting is to send information to receiving units as a simple list of environmental parameters of interest during a specific period of time. This sort of support can take many forms, it can be sent via naval message traffic, IRC chat, or even classified or unclassified e-mail, and is usually re-issued on a regular hourly interval as per the request of the customer. It is not so much the form or regularity of this type of forecast that is of particular interest. It is, instead, the assertion that these forecasts include no interpretation of the environment, but rather a detailed description of it. This requires the forecaster, who is often well outside of the operating area, to generate a product based on a point latitude/longitude position and time period. The recipient then has to read the message and put it into context by considering the impact of the forecasted conditions upon the type of operation planned for execution. This format allows the Navy to consolidate its forecasting assets into the geographic METOC Centers that are in existence today. Such efforts can reduce the manpower required to support units spread over a large region and, by extension, also reduce the budget and logistic footprint necessary to do so.

The drawbacks to sending forecast information as a list of parameters from a remote location, however, are copious. First, individuals who may be hundreds, or even thousands, of miles away, are producing these forecasts. This may result an unclear understanding of locally produced, microscale phenomenon. This distance from the customer can also make relaying information or updating the operational picture extremely difficult. As an example, if a customer needs to change the type of operation they will be conducting on short notice, there may not be enough time to relay this information to the nearest METOC Center. As a result, the operators are left with a forecast that is of either little or no use to them because they have changed location, timeframe, or asset employment. In this type of situation, the meteorology professional may also not be aware of when they should amend their forecast due to quickly changing conditions. This could result in a loss of their credibility from the viewpoint of the

customer. The onus for determining the impact of environmental parameters is placed entirely on the receiving unit, while the forecaster is relieved of the responsibility of making such assessments.

Another manner in which forecast information is provided to customers is as "impact-centric" information. In this approach, the METOC community member is made aware of what type of assets or platforms will be used during an operation. Then a forecast is produced in a similar manner as before, but an additional layer of context is applied to the problem by the forecaster as they consider how these conditions will impact the equipment that will be used. An example of this type of product is a "stoplight" graphic. These use a red, yellow, or green color code to represent the severity of degradation an asset will experience as a result of environmental conditions. The color code is assigned by the meteorology professional based on the type of platforms planned for use, their safe operating thresholds, and the forecasted conditions. It should be noted that this type of assessment can also be produced from a regional METOC Center, and therefore incorporates the benefits and deficiencies previously mentioned. In addition, the forecaster must now have an accurate understanding of exactly which types of equipment will be used. If another type of asset is substituted, an assessment may not be immediately available.

Another example of a kind of "impact-centric" assessment is the TDA. These are software units that have been designed to specifically evaluate the impact of environmental conditions on a particular kind of operation employing distinct assets. Generally these lack flexibility since, by definition, they are geared to solve only one type of specialized problem. Since these modules are generally smaller, however, they are usually more transportable and a non-METOC individual can be trained to use them if necessary.

A third method of providing forecast information in support of an operation is through the assignment of an attached METOC forecaster. In this situation there is either an officer or enlisted service member that is traveling with the commander and is on-scene to provide immediate updates to the environmental picture. The forecast is still produced using numeric model output as before, but now an individual is present with the

customer to incorporate any type of context they may desire and evaluate the information in that manner. Customer response to this kind of response is usually overwhelmingly positive, since it is an augmentation of their manpower and they can work closely with that assigned individual to tailor the support in whatever way they desire. Practically, however, this means of providing environmental services is manpower intensive, logistically intricate, and a strain on monetary resources, which makes it undesirable except in isolated cases.

C. PROCESSING DATA AND PASSING INFORMATION – THE WAY OF THE FUTURE

In an article he wrote as the Director for the Office of Force Transformation, Vice Admiral (ret.) Cebrowski defined transformation as "a continuing process… [that] is meant to deal with the co-evolution of concepts, processes, organizations and technology. Transformation is meant to identify and leverage new sources of power." (Cebrowski, OFT). In a presentation he gave on 24 February 2004, CAPT Chris Gunderson expanded upon this statement by saying that transformation was the utilization of technological advances within our current "information age" combined with increasing efforts to expand globalization (Gunderson, 2004). These principles are uniquely suited for immediate application within the METOC community in terms of how we process meteorological data and exchange that information with our customers.

Figure 1 illustrates how transformation can develop by combining technological advancements with increasing interoperability. This diagram depicts advances along the globalization axis increasing from left to right, and advances in information age technologies increasing from bottom to top. The examples of globalization range from combining inter-service meteorological numeric models to the utilization of coalition forces. Advances along the information age axis increase from the use of software agents to the employment of wireless networks and advanced microscale modeling. The advances along each individual axis meet along the dashed line, which represents progress towards transformation, which in this case is applied to the field of Navy meteorology. These are examples of technologies that are either available now or in development, and would only require an open, innovative mindset and effective leveraging in order to make them a reality.

The handling of data has long since moved from the realm of simple storage medium into a more advanced representation of how that data may interrelate and come to represent information. An example of this would be the development of relational databases, which sought to do more than simply store the data they contained. In a similar

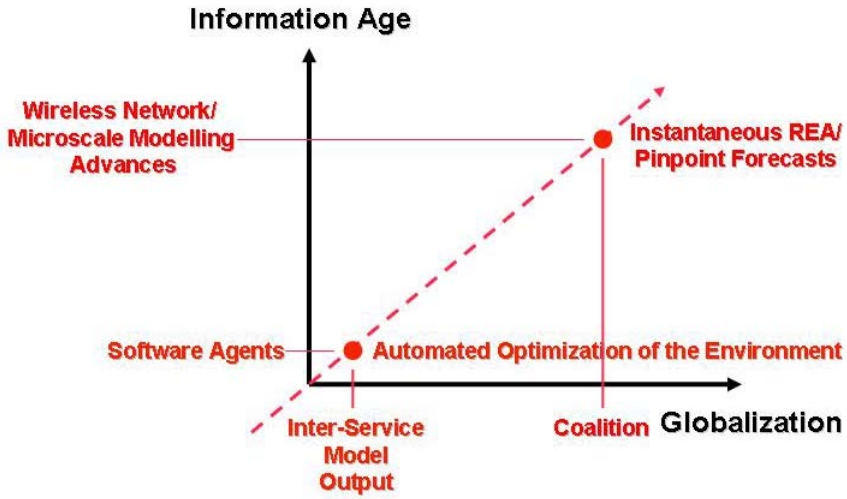

Figure 1. Working Towards Transformation.

way, other advances in how data is represented and processed have become more focused on the context of data and automating interpretation. One such advancement has been the development of software agents. Though they will be examined in much greater detail in the next chapter, software agents are essentially autonomous modules that can be programmed to perform various tasks that could include data manipulation and interpretation. This type of automation could potentially streamline the process for producing a forecast, depending on the format of the final product and the levels of context that must be applied to generate it.

Advances in how information is passed that are independent of hierarchical stovepipes are an example of increasing globalization. There can be no doubt that as information continues to travel around the Earth at an increasing rate, globalization is happening. Large quantities of data or information can now travel from one point to another almost instantaneously at the touch of a virtual button. If there is any problem with receiving and using that information it is usually because of formatting, which may

be dependant on a particular piece of software or interface. This highlights the importance of creating a common architecture for passing information that will also serve to enhance its usability wherever it is accessed.

D. THESIS APPROACH AND GOALS

This thesis describes solutions for the METOC community that address the two means of advancement proposed above. The first portion of this thesis will consist of coding a software agent in the programming language Java. This will serve as a fresh look at how meteorology data is processed within the Navy, and evaluate whether or not automation in this area is something that would be possible or desirable. The second portion of this thesis will encompass the passing of data or information that has been output by the developed agent. This will essentially be the examination of a new approach as to how to standardize the means by which exchanges take place. Ideally, this new architecture will be a means by which organizational stovepipes can be either bridged or eliminated. For research purposes, this thesis will examine how the developed agent passes information to, and receives information from, a piece of decision aid software. Specifically, this software will be the SEAWAY program developed by CDM Technologies, Inc of San Luis Obispo. The SEAWAY program will be detailed for the reader in a later chapter.

The expected results for this thesis are twofold. The first anticipated result is that the creation of this agent will enable the design of a new architecture for passing information within the METOC realm. This innovative architecture will be proposed and then evaluated for viability. The second is that the successful example of a software agent that is specialized for use within the METOC community will allow for a detailed examination of the feasibility of this type of programming, as well as a critique of its potential impact. This resulting impact assessment will also highlight the need for the METOC community to effectively document and define its functional requirements for the emerging FORCEnet environment.

THIS PAGE INTENTIONALLY LEFT BLANK

II. AGENTS

A. WHAT IS AN AGENT?

Interestingly enough, the properties that exactly define an agent are a point of disagreement among computer scientists and artificial intelligence researchers. There are as many different listings of the components of this type of software entity as there are researchers in the field. So, in order to begin a discussion on this topic and apply it to the research discussed here, a general definition must be presented and agreed upon. In his book on multiagent systems, Gerhard Weiss (Weiss, 1999) defines an agent as "autonomous, computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors". He also stresses that, "there is a general consensus that *autonomy* is central to the notion of agency". These two basic definitions serve as strong starting points for our discussion and, as we will see later, provide the framework for defining the program within this thesis as an agent.

Despite the fact that there are many ways in which software agents have been defined, virtually all scholars agree that agents to some extent share three common characteristics. The first of these characteristics is a means by which the agent can receive stimulus from, or sense changes in, the environment. This serves as a source of input for the software entity and allows it to receive information for processing or prompting from its surroundings. The second characteristic that most agents possess is the presence of some sort of internal environment. This is a very general definition of whatever rules or methods govern the actions of the agent. This is what defines how the agent will process stimuli and how it will determine what the appropriate response is within its environment. The final common characteristic is the means by which the agent acts upon the environment. Once an agent receives information or stimulus and decides what course of action to take, if any, is how it influences and acts upon its environment. These three components can be summarized as an input mechanism, an evaluation process, and output mechanism. The evaluation process characteristic, or the internal environment of the agent, will be discussed further in the following section of this chapter.

B. TYPES OF AGENTS

There are two main, general categories of software agents. What separates these categories from each other is the complexity of the internal structure of the agent. In reactive agents the internal structure of the agent is relatively simple, while in a cognitive agent more complex processes are incorporated.

1. Reactive Agents

This type of entity is the most basic form of the software agent. As the name implies, reactive agents are designed to perform specific, simple actions as a result of interaction with their environment. In most cases, these actions are triggered by key events, after which the agent uses a simple set of rules to determine the proper response. A stimulus is applied to the software agent and a predictable, foreseeable reply or reaction occurs. This response is a regular occurrence that will always be the same for a specific causation.

The internal structure for this type of agent is usually a set of rules or a list of actions. As a stimulus condition is met or an event occurs, the agent processes the action by running through this list of rules. The agent then uses a specific rule to determine the appropriate response, if any apply. If none of the rules apply or the necessary conditions are not met, then the agent may perform some sort of a default action, or take no action at all.

An example of a reactive agent would be a simple solution of the El Farol programming problem. In 1994, Brian Arthur introduced a programming problem that consisted of trying to model the decision process of a patron deciding whether or not to spend an evening at the El Farol Bar in Santa Fe, New Mexico (Arthur, 1994). Essentially, the problem statement consisted of a bar whose patrons were happiest when an optimal occupancy was achieved. For example, if the ideal occupancy of the El Farol were 60 people, the patrons would be unhappy if they arrived to find 100 people crammed into the small tavern. Likewise, customers would be unhappy if they did not go to the El Farol for an evening of entertainment, and then discovered that only 30 people had been present at the bar. In this second case, the patron could have easily gone to the bar and had plenty of room to be comfortable. A reactive agent would only focus on comparing the actual attendance with the ideal attendance and would use a simple rule, or

set of rules, when deciding whether or not to attend. For example, a reactive patron agent could be keeping track of the weekly attendance at the El Farol for the last five weeks. If the average attendance for the bar during these weeks were 70 people, the agent would use a simple comparative rule to "decide" to not go to the El Farol this week. An obvious limitation of this type of simple solution to this problem is that the agent is not very flexible in responding to external stimulus. The response is determined beforehand by a rigid set of rules that would apply in exactly the same way in every similar instance.

2. Cognitive Agents

This type of entity is a much more advanced form of software agent. Cognitive agents are, by their nature, designed to model more complex objects than reactive agents. These pieces of software model the decision making process by attempting to model cognitive thinking. These types of agents react to their environments in a more versatile manner, and a specific stimulus may result in different responses depending on the context of the initiating events.

The internal structure for this type of agent is usually centered on a system of goals. These goals, or sets of goals, are designed to model the purpose or intent of the modeled object. It is essential to cognitive agents that there be some sort of objective that the entity is trying to achieve. Along with this, the agent is also designed with a measurement system of some kind. This measurement allows the agent to evaluate the progress it is making toward achieving those goals. It also allows the agent to weigh goals against each other and to possibly choose between actions in order to resolve conflicts between opposing goals.

An example of cognitive agents would be software designed to model some of the properties of a bird. In 1987, Craig Reynolds designed a programming problem that tried to emulate some of the characteristics of animal flocks, herds, and schools (Reynolds, 1987). Since Reynolds worked on this program while in New York and applied it to flocks of birds, the problem became affectionately known as the "Boids" problem (pronounce "birds" with a Brooklyn accent). Though this problem can take many forms, a common one consists of designing bird objects that have several goals and makes decisions based on the urgency of achieving those goals. An example of this may consist of programming 'Boids that have three main goals, Flocking, Circulating, and Collision

Avoidance. The flocking goal would drive the ‘Boid to seek out and gather with other ‘Boids so it would not expire from loneliness. The Circulating goal would prevent the ‘Boid from flying around in circles so it would encounter new scenery and would not die from boredom. The Collision Avoidance goal would, obviously, be designed to prevent the ‘Boid from hitting obstacles and injuring or killing itself. The measurement or evaluation of these goals would then allow the ‘Boid to determine which goal was most urgent, and make decisions accordingly. For example, if a ‘Boid was flying in a flock while achieving good circulation it would be satisfying two out of three goals. If the flock turned toward an obstacle, however, the Collision Avoidance goal would become more critical as progress toward that obstruction continued. At some point, that ‘Boid would have a decision to make. On one hand it would have the criticality of satisfying the Flocking goal by staying with the other ‘Boids despite their impending doom. On the other, it would have satisfying the Collision Avoidance goal by turning away from the obstacle and possibly separating itself from the group. In this way, simple cognitive objects are created and then turned loose to interact with the environment and make independent decisions.

Extending the definition of the cognitive agent could include the use of programming tools such as genetic algorithms. This type of encoding would allow certain traits to be passed on to different generations of software agents, and allow an agent population to evolve and refine itself. Certain traits or dispositions could be programmed into various versions of a similar agent, or be allowed to develop during a breeding cycle. For example, a ‘Boid could contain a genetic value that determined how close it would come to an obstruction before turning. We could also program our environment so that the ‘Boids would breed with a partner every 100 movements, produce two offspring, and then die. The resulting “closeness” value for the offspring would be an average of the two parents. It is very likely that in this case, after a certain number of generations, low “closeness” values would be completely bred out of the population. ‘Boids that got too close to objects before turning would eventually keep colliding and dying and, thus, be eliminated from the genetic pool.

C. WHY USE AGENTS?

Software agents are programmed entities that are uniquely suited for effective employment in the examination of meteorological data. As Dr. Jens Pohl noted in his paper on applying context to create information (Pohl, Transitioning from Data to Information), we are in an age where users can be overwhelmed by the vast amount of available data. This description accurately encompasses the enormous amount of meteorological numeric model output that METOC professionals have available to them. Software agents are extremely well suited for sorting through this kind of output using either reactive or cognitive processing methods. Reactive agents could handle considerable quantities of model output and perform tasks as simple as highlighting locations where wind speeds exceeded a certain value. This capability is currently available in METOC data retrieval and assessment software, but agents could build greatly upon this utility in many ways. Given a list of platforms that are to be used during an exercise, reactive agents could also compare forecasted conditions with operational thresholds. Cognitive agents could extend these benefits by applying several goals to model output processing. In a more advanced iteration, these kinds of agents could use genetic algorithms and model verification to track which models are most accurate in certain geographic regions. They could then produce composite forecasts by combining portions of different model forecasts. This possibility becomes especially significant when considering the speed and relative ease that software agents would be able to process vast amounts of data.

In and of itself, raw data is rarely significant to the warfighter. Dr. Pohl stated in the above mentioned article that transforming data into information and knowledge that can be leveraged as an asset by the user is where the true power of computing can have the greatest impact. Initially, this procedure would begin as the simple loading and processing of data according to a set of predefined, user-entered rules. In theory, depending on the complexity of the agent's data manipulation rules, these software entities could eventually begin to find relations between various pieces of data. As these relations are recognized and developed, context is created and the data becomes information. Dr. Pohl presented the use of ontologies to create this context, but a

carefully crafted reactive or cognitive agent could perform a similar function and produce similar results that would be significant to an educated user.

D. APPLYING AGENTS TO THIS THESIS

In the spirit of learning to walk before trying to run, reactive agents are examined within this thesis. The agent that will be developed will serve as a link between numeric model output and command and control software and will be referred to in this paper as the Java Version Weather Agent, or WAg for short. The next chapter will detail the command and control tool that will be examined and used as a sample platform from which to develop the requirements for this agent. The WAg itself will be outlined and it's development detailed in the fourth chapter.

Though not coded or applied to research directly here, cognitive agents and the use of ontologies to automatically produce meteorological information and knowledge from numeric model output bears consideration as well. This topic will be discussed as appropriate within this thesis.

III. SEAWAY

A. WHAT IS SEAWAY?

The SEAWAY program is a project developed by CDM Technologies, Inc of San Luis Obispo. CDM Technologies began as an offshoot of a student research lab within the Computer Science Department of the California Polytechnic University. As this group began to pursue research into expert-agent software, specifically incorporating the application of ontologies, interest in producing decision-aid tools using this technology began to develop. This group started to program software agent solutions for both government and commercial organizations. As finished products were fielded, it became clear that providing adequate follow-up support for this software was a sizeable commitment that surpassed the resources of a student-staffed group. This led to the creation of CDM Technologies, which still has strong ties to Cal Poly through former students and interaction with faculty such as Dr. Jens Pohl. CDM describes itself as specializing in developing "decision-support tools that promote effective planning and coordination of complex logistic operations" using "expert-agent technology". This chapter will be a general introduction for the reader into the capabilities of SEAWAY, and is not intended to serve as a user guide for the program or provide the specific detail that is commonly known as 'buttonology".

According to its sales brochure, SEAWAY is CDM's "latest decision-support system designed to satisfy the focused logistic demand of Joint Vision 2020". It also defines SEAWAY as a, "joint decision-support system for sea base logistics planning and coordination". This program is essentially a command and control software tool funded primarily by the Marine Corps and developed to support logistics operations and track requirements. CDM advertises three main support areas that this software is specialized for. The first of these is to assist in Cargo Visibility. This essentially consists of taking all logistical assets within a theatre, and making them visible to high-level planners. A second mission area is Cargo Operations. In support of this, SEAWAY can tell it's user on what ship, and in which hold, an exact supply component lies. This allows for detailed tracking of precisely where replacement parts or supplies are located. The third mission area SEAWAY supports is Mission Planning. In this respect, the software develops and

updates offload plans that commanders can use to project the logistical requirements of ashore units. The expert agents within the program predict the requirements of employed units based on their particular scheme of maneuver. A fourth mission area is Mission Tracking, which allows the user to see continuous updates of mission status and logistical standing during an operation. The final mission that SEAWAY was designed to support was Mission Execution. In this regard, the versatility of this program allows commanders to examine various execution options as the mission is underway and unfolds. By providing mission support for the aforementioned five areas SEAWAY establishes itself as a powerful planning tool for warfare commanders, which is one of the reasons it was chosen for use along with the development of the WAg.

The overall SEAWAY program actually consists of three main components, the server, the agents, and the client. The server acts as a means to load or save archived information and define the paths that the program will use to access information. It can initialize the SEAWAY program so that users will have access to either previously defined areas and scenarios or the default data. The agent element of this program coordinates and manages the interaction and initialization of the different kind of agents that SEAWAY uses. The types of agents this program incorporates are requirements, delivery, inventory, weather, movement, route, tactics, and siting. It is important to point out here that even though SEAWAY already has an existing weather agent, the agent this thesis is researching is different for significant reasons. SEAWAY uses its weather agent as a means of evaluating how a forecast will impact a programmed logistics delivery route. At this point, however, the program requires the user to enter the weather parameters by hand before any kind of evaluation can be made. The WAg is intended to automatically produce forecast arrays directly from numeric model output, which would provide utility for this information to be imported directly into command and control software such as SEAWAY. The user interfaces with SEAWAY through the client portion of the program. This is the window that displays the operation area maps and provides access to all of the program menus. This is what a typical user will think of when asked about this program. The three main components of SEAWAY are all important, but in particular the agent element provides special utility that will be further examined in the next section of this chapter.

B. WHY USE SEAWAY?

The CDM Technologies program SEAWAY has several features that make it well suited for interaction with a military agent in general, and this planned WAg in particular. A basic characteristic of this program is that it is coded in the computer language Java. In general, this is a versatile language since it is largely platform independent, meaning that it can run on any kind of platform as long as it is loaded with the latest version of the Java Virtual Machine (JVM). The WAg this thesis is researching is programmed in Java as well, which is convenient since it minimizes and problems that may have developed due to language-specific differences. In general SEAWAY is also a good program for military use since it imports military standard maps. The software displays and allows notation on National Geospatial-Intelligence Agency (NGA) maps. This is a desirable feature since military operation might be planned using a similar map or standard. In addition, this program can also import digital terrain elevation data (DTED) maps, although these are generally not preferred due to the excessively large size of these files. There are several other reasons that SEAWAY was examined in conjunction with the development of the WAg that will be detailed in the next chapters.

In the previous section it was mentioned that the SEAWAY agent element would be revisited in more detail. This component of the software, which manages the program's agents, is actually a key feature that facilitates the incorporation of the WAg. This allows the program to create an instance of the WAg and all of its components. SEAWAY then also has access that enables it to instantiate the request and forecast objects that are a part of the WAg. This permits the direct exchange of information and avoids other more complex methods of passing information to and from SEAWAY. Without this type of utility, exchanges of data would have to use a complex application programming interface (API) and proxy object wrappers. The agent feature of SEAWAY greatly simplifies the integration of the WAg with this program.

Another characteristic of SEAWAY that makes it a good choice for use with this thesis is that it already incorporates some consideration of environmental factors. This software was designed from the beginning to consider all aspects of logistics, including factors that could impact the delivery of supplies such as hazardous weather. As a result, SEAWAY comes complete with weather symbology that can be displayed on the NGA

maps. Additionally, this program can also evaluate the impact that specific weather parameters will have on logistic delivery routes. This leaves the WAg free to focus on the specific problem of producing and providing forecast information from model output. All of the characteristics mentioned within this chapter make SEAWAY a good choice for use when examining the proof-of-concept creation of a WAg program.

IV. THE WEATHER AGENT (WAG)

A. THE PURPOSE

One of the main reasons the concept of a WAg was chosen for examination within this thesis was to evaluate its feasibility as an element of FORCEnet. Two of the tenants of FORCEnet are that government technology, specifically technology used within the Department of Defense (DOD), should be independent of "stovepipes" and utilize web services. The use of agent-based software is a good choice to meet both of these provisions, and can be a resource the METOC Community can implement in order to transform the means by which it provides service to its customers.

A "stovepipe" is an organizational configuration that bears a visual resemblance to its namesake due to an inherently restrictive scope that serves only to meet a specific need. This term is most commonly applied to command structures that are narrow, and only support communication between junior and senior components. While this may facilitate the flow of information up and down the chain-of-command, it restricts lateral movement and the sharing of ideas between component commands within the organization. This view can be applied to naval communities, the branches of military service, or to branches of government service. This kind of restrictive organizational structure is also a problem when considering how technology is employed within the DOD. When an individual member within the Department takes steps to meet a technological need it is often done by contracting a commercial agency to develop a specific piece of software utilizing proprietary knowledge. This creates a problem since that proprietary knowledge often cannot be shared between commercial sources and, as a result, the various ensuing products are then not compatible.

Agent-based technology can help rectify this situation by encouraging the production of common architecture software that is produced in accordance with a specific standard. For example, the WAg is programmed in the platform independent language Java, is intended to be an open source product, and will serve as a bridge between a government produced numeric model and a privately licensed command and

control tool. Until an ideal FORCEnet operating environment is achieved, agents can be utilized in this way to breakdown technological "stovepipes" and become a means of enabling interoperability.

The web services aspect of FORCEnet would enable technological assets to remain distributed, even forward deployed, but still capable of reaching back for support on an as-needed basis. A focus on providing support via web services has the potential to allow DOD customers access to information at any time. For example, if a naval Commander Amphibious Task Force (CATF) and marine Commander Landing Force (CLF) were coordinating to plan an amphibious landing, current weather information would be vital to safe and effective operations. Fully integrated FORCEnet connectivity would allow these commanders to reach back to the nearest METOC Center and download this information as required. Moreover, a full and complete implementation of FORCEnet would keep providing updates of that environmental information as it became available and would allow full integration with display capabilities. This would make it possible to project that weather data within whatever planning tool was already in use. Agent-based software is a programming methodology that could enable exactly this type of support. Aside from breaking down technological stovepipes, as mentioned in the previous chapter, programmed agents can be implemented to take advantage of this type of web service support.

As an example, the WAg allows users of SEAWAY to request weather information via a local agent with the push of a virtual button. The WAg then uses an Internet connection to contact servers at a weather center, in this case Fleet Numerical Meteorology and Oceanography (FNMOC), and utilize available web services. These same agents could be set so that they would continue to poll the FNMOC servers at a set interval based on model runtimes to ensure that the delivered data remained updated. In other cases, software agents on FNMOC's servers could likewise push data to various customers at set intervals depending on the type of request. Agent-based software could be a reliable means of implementing the web services tenant of FORCEnet.

B. AGENT REQUIREMENTS

Before beginning to design the WAg it is important to clearly define what will be required of the software. Once we outline exactly what the requirements of the software

will be, we have a set of goals to work towards and a yardstick by which to measure the success of this effort. In software development there are two kinds of requirements, functional and non-functional. In his book on software engineering, Stephen Schach (Schach, 2002) defined these two classifications in the following way, "Functional requirements relate to the functionality of the target software…Non-functional specifications specify properties of the target software, such as reliability and maintainability, or relate to the environment in which the software must run". A useful exercise would be to consider the types of requirements you would be concerned with if you were constructing a deck for your home. A functional requirement would be specifying the use of three-quarter inch bolts to fasten the joints. A non-functional requirement would be stating that the deck should be weatherproofed so it can withstand wear and tear from rain and sun. This section will break down the requirements for the WAg into these two categories.

1. Functional Requirements

The first major functional requirement that had to be defined was the types of meteorological or oceanographic products the WAg would process for SEAWAY. As previously mentioned, the designers of SEAWAY had built certain environmental parameters into their software. In their design process they used an unclassified Navy weather forecast and, as a result, had incorporated parameters such as sea state, winds, etc. Since this thesis was intended to be a proof of concept attempt at implementing agent technology in this manner, it was important to have a scope suitable for an initial effort. Thus, it was determined that the agent would only be required to return a few forecast parameters vital to surface operations within a littoral environment. This information would include surface winds, significant wave height, and 12-hour precipitation amounts. In addition, surface wind would be further defined as the wind 10 meters above ground level. These were also convenient choices for forecast parameters are easily accessible from most meteorological or oceanographic numerical models in a gridded format.

The second major functional requirement that had to be defined was which numerical model would be used as the source for downloading this gridded output. Since this is a thesis being researched at a military learning institution, it was easy to decide to use models developed by the Naval Research Laboratory (NRL) and National Oceanic

and Atmospheric Administration (NOAA). These models are in wide use within the Navy, and the author had practical forecasting experience with them as well. Considering that this project would focus on wind output, 12-hour precipitation amounts, and significant wave height, the Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS) and the Wave Analysis Model (WAM) seemed to be reasonable selections. As the name implies, COAMPS is a coupled air/ocean model. The website of FNMOC describes it as a model that runs in a continuous update cycle and gets its initial fields from the Navy Operational Global Atmospheric Prediction System (NOGAPS) model grids using real and synthetic observations. It calculates wind components, potential temperature, mixing ratio, surface pressure, ground temperature, ground wetness, and sea surface temperature. The grids are latitude-longitude or Cartesian coordinates on a horizontal map projection and can calculate parameters for up to 30 staggered vertical levels. It incorporates 1-kilometer database topography from the Defense Mapping Agency level 1 DTED, which can significantly impact near surface parameters. WAM is a global model that produces 1-degree resolution gridded output for various oceanic parameters. These include significant wave height, mean wave direction and period, secondary wave direction, secondary wave mean period, whitecap probability, and maximum wave height as well as direction, height, period for both seas and swell.

After deciding which parameters and models the agent would utilize, it was important to decide what data service the WAg would use. FNMOC runs several data servers and subscription services that include the Tactical Environmental Data Server, the Metcast server, and the newly developed Data Blade server. For the sake of simplicity, the Metcast server was chosen for use. This was also chosen due to the author's familiarity with the request procedure for this service. It is important to note, however, that conceptually the architecture of the WAg can be applied for use with any data service. This flexibility is one of the major strengths of implementing agent-based technology, and highlights the versatility and future potential of this thesis topic. TEDS services or the Data Blade could both be utilized by the WAg in future iteration of this project. According to the Metnet homepage, Metcast is a subscription service that runs on an application server. It receives requests via hypertext transfer protocol (http) or hypertext transfer protocol secure (https) that are formatted in the Metcast request

language, MBL. It can return gridded, observational, imagery, or text products. Metcast provides access to the COAPMS and WAM model output required for use by this agent. Even this basic description highlights how Metcast meets many of the service requirements of the WAg.

Another functional requirement that had to be decided before beginning the production of this software was how information would be passed between the agent and SEAWAY. Since the SEAWAY program could create a local instance of the WAg, this decision boiled down to the preferences of the SEAWAY developers for the forecast parameters. During an initial discussion it was mentioned that within their program they would be converting many of the internally held values to parsed integers, to include latitude and longitudes. Wind speeds and direction, significant wave height, and precipitation amounts can all be adequately represented using such integers. These three values would then be combined in individual arrays with their corresponding geographic locations.

2. Non-Functional Requirements

A major non-functional requirement for this software is that its development be grounded in common standard architectures. As previously stated, this would prevent the agent from being dependent on proprietary information that might restrict its ability to interact with other pieces of software. As an example, this would include being able to request and receive information via http or https. This is the protocol used to transfer information across the Internet and is internationally recognized. The software should be platform independent and be able to run on any type of operating system. This was also previously discussed in this chapter as the reason it was programmed in the Java language developed by Sun Microsystems. In general, this software would be designed in compliance with the guidance provided by the International Organization for Standardization (ISO). The ISO is an international organization whose purpose is to standardize software-programming guidelines in an effort to ensure interoperability between computing systems.

The size of the agent and the delivered data are also worth considering as a non-functional requirement. Since this agent is intended to be a portable part of a distributed system, the size of the WAg itself must be kept manageable. If the agent became too

large or unwieldy, it might interfere with the command and control software it interacts with. This kind of effect would be contrary to the benefits the agent would provide to the user. The size of the downloaded gridded output must also be kept manageable and within reason. Since much of the operational planning that occurs takes place in forward deployed locations, available bandwidth must be viewed as a valuable commodity. In many areas Internet access is inconsistent and bandwidth-limited, which would prevent the transfer of large files to and from the WAg. This will be discussed further in the following section, but was kept in mind when writing the program code.

C. THE AGENT STRUCTURE

There were several different ways to structure the agent that were considered when writing this software. It was important to evaluate how the agent would be configured internally, as well as how it would be placed within its environment. As we will see, how the system was structured would have positive and negative implications that had to be considered during development. As a beginning point, however, the commonly accepted general structure of agents was useful in deciding how to begin defining the architecture of the WAg itself, prior to placing it within an environment.

The basic definition of an agent is as a software entity that has an input, an output, and some sort of an internal environment. This simple description establishes a means by which we can begin to structure the WAg. Since the agent would be handling requests from the planning software for forecast parameters, this obviously becomes the input for the software. In order to produce forecast information we must provide the Metcast server with several pieces of information. These include the latitude-longitude bounding box, the desired timeframe, and the specified desired environmental parameters (which in our case has already been determined to be surface wind, significant wave height, and precipitation amounts). These items can be grouped by considering them as items that together compose a request. In turn, this request can be instantiated within the Java environment as a software object that has attributes that correspond to the grouped items. Other attributes that may be significant are the assignment of a request identification number, the assignment of a forecast number, a binary value that indicates whether or not the request has been fulfilled, and some sort of value that denotes the desired granularity

or resolution of the model. In this way the idea of a request object, that can be understood by the agent and will be created by the planning software, is formed.

Since the agent will be returning a forecast to the command and control tool, this becomes the obvious choice for the output of the software. In the case of this thesis, where we are only investigating feasibility via the processing of selected parameters, this object is simplified considerably. In order to track the returned or issued forecasts it is reasonable to include some sort of identification number as an attribute of the forecast object. Aside from this, the forecast object must also contain the forecasted parameters. In the case of surface winds, for example, this particular attribute will consist of an array of values. From the requirements we see that these will be parsed integer values for the latitude and longitude, and integer values for the wind speed and direction. Together, these items would compose the output of the WAg.

The internal environment of the agent is composed of several main components. The first is how to receive and handle the request object. Since this thesis uses the Metcast server to return forecast values, at some point the request object must be converted into MBL. As the geographic location of interest was changed, or as the desired forecast parameters were changed, the agent would have to change the structure of the Metcast request accordingly. The fact that this thesis is a proof of concept look at this topic allows us to simplify this by "hardwiring" the agent with request language for our three pre-selected values. After the request language has been defined, the agent must establish an http connection with the Metcast server and deliver the request. As that data server receives and responds to the request, an input stream must then be initiated to capture that data. Finally, the captured data must be processed in such a way so that the forecast object array is created and ready for delivery.

D. THE STRUCTURE OF THE ENVIRONMENT

Once the architecture of the agent itself was established, how it would be placed within its environment was a topic worthy of considerable thought. If we conceptually understand the desired effect of FORCEnet, we can envision how the agent would be free to interact with any planning software. With access to data servers on which to run the WAg, it might also be desirable to structure the environment to incorporate this capability. The way in which the agent fits within its environment can also be simplified

so that the work on this proof of concept attempt at implementing this idea is streamlined. Realistically, each of these individual possibilities represents a progressive step toward the realization of an overall fully integrated computer architecture.

In a perfect FORCEnet environment, the WAg would be able to exchange information with any command and control software through pre-established interfaces. This sort of view is presented in Figure 2. Here, SEAWAY would be able to create request objects on its own and initiate an http connection directly with the WAg. The

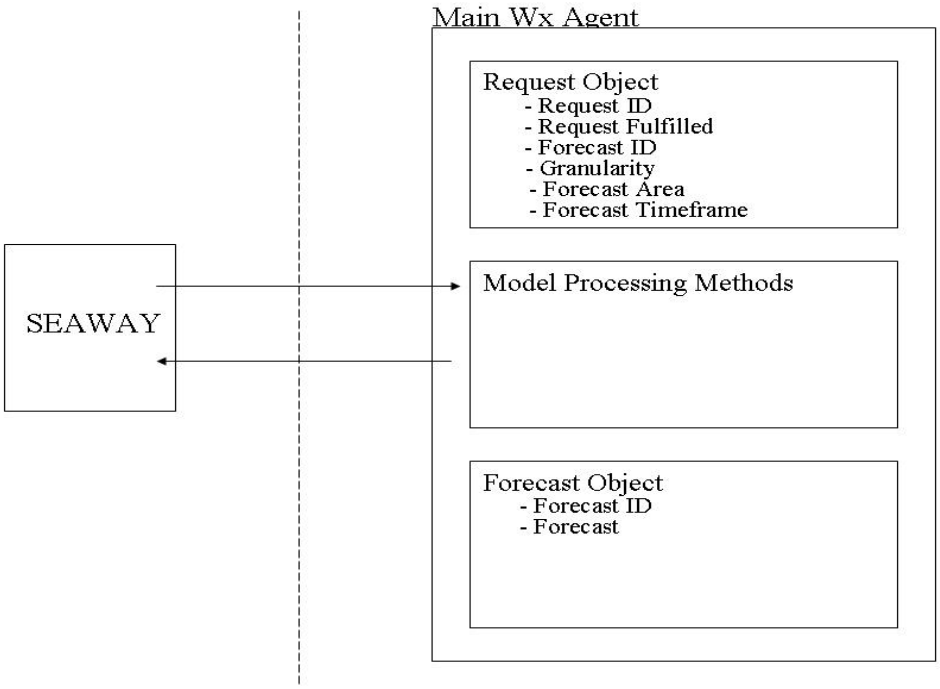

Figure 2. The WAg in a FORCEnet Environment.

agent, in turn, would be running on a centralized server. This would allow the agent to receive multiple requests from different warfare planning tools, as permitted by the allocated bandwidth and computing power. The agent could then receive model information at that centralized location, process it and return forecast objects as desired by the requesting software. The forecast objects could then be imported directly into the planning software and processed immediately. Ideally, once it had received forecast information the planning software would also be able to display the appropriate symbology for that forecast to its user via its graphical user interface (GUI). If the

command and control software was also configured to incorporate threshold information for operating platforms, it could also use its GUI to indicate the effect the forecast conditions would have on the planned exercise. It is fairly easy to see that this sort of architecture would have several advantages. First, there are less “moving parts” to this sort of structure, since the planning tool communicates directly with the WAg. Second, the agent is running at a central location, which allows it to track and respond to requests from differing locations. Third, the overall bandwidth required for communication is reduced since only the request and forecast objects are transmitted instead of entire model runs.

Since we do not yet operate in the ideal FORCEnet environment just mentioned, we must consider other means of integrating the WAg with its environment. The main area of difficulty is found at the interface with the warfare planning software. Without integration during development, these pieces of software will have no way of creating the request object that can be sent to the WAg. Moreover, they will have no way of deciphering the returned forecast object into meaningful and useful information. In order to bridge this gap the command and control tool must have some local way of instantiating these objects and passing information back and forth. This was a significant problem encountered when working with SEAWAY.

Computer software usually passes information back and forth through an application program interface (API). An API is a sort of gate through which data, or information, must be passed in order to import or export it. After meeting with the CDM Technology developers it became evident that, in their own words, the use of the SEAWAY API required complex bundling and packaging in order to pass data. The architecture of the SEAWAY program itself, however, provided a way of bypassing this problem. As stated in a preceding chapter, the SEAWAY program uses a manager entity to create and run the instances of its agents. This would allow the creation of a scaled-down, local version of the WAg that could be instantiated by this manager. Since SEAWAY would be creating the instance of this local agent object it would then be able to use and understand the request and forecast objects. This structure is diagrammed in Figure 3.

When designed in this manner, as the WAg creates a request object as per the instructions of the planning tool, the local agent then establishes an http connection with a full version of the agent running on a data server. The request object is passed, the appropriate model information downloaded and processed, and then the forecast object is created and returned. As with the previous design, this sort of structure keeps the required bandwidth low and allows for a centralized processing of the requests. In this case, however, there is some added complexity in that an additional level of communication is required between the local WAg and the command and control tool. Additionally, this

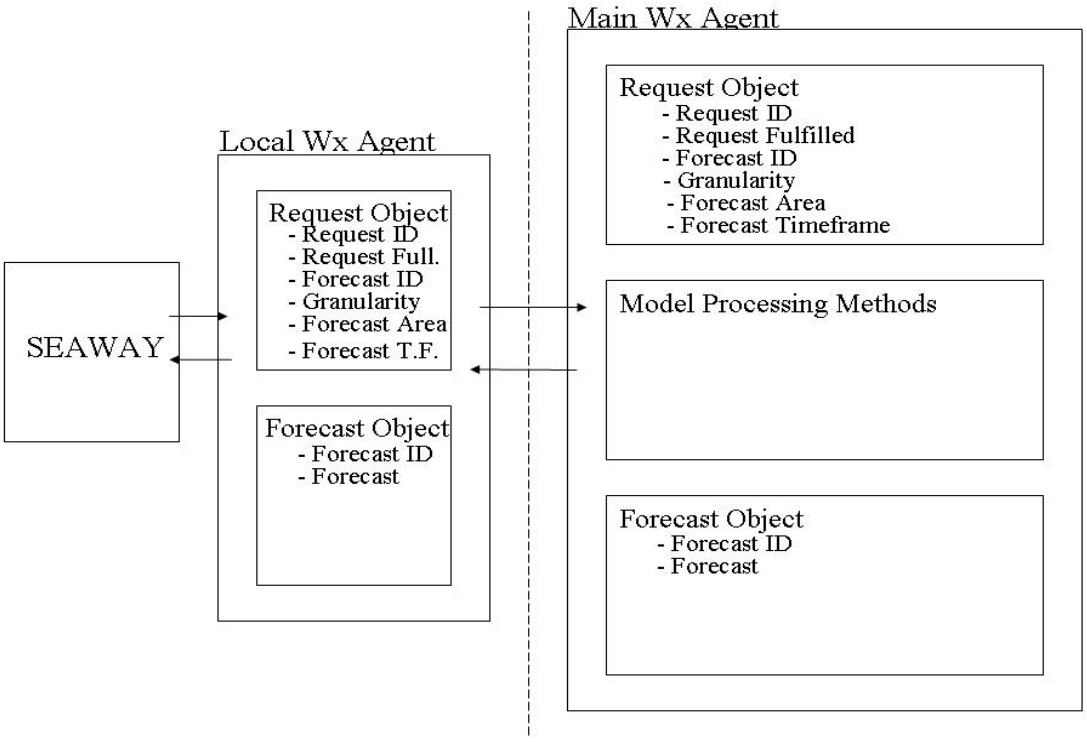

Figure 3. The WAg Bridging the Proprietary Information Gap

communication problem would have to be solved for every different type of planning software that the agent would interact with. This sort of design structure could be viewed as a way of solving the interoperability problem until wider implementations of FORCEnet were in place.

The final structure of the agent environment we will examine is the one that was actually implemented within this thesis. During the course of this research it was not

always practical to plan on running a full, or main, version of the WAg on a data server remotely located from the Naval Postgraduate School. As just discussed, however, SEAWAY can initiate and manage agents locally. With this in mind if SEAWAY simply instantiated the full WAg locally instead of a smaller version the results would essentially be the same. Figure 4 outlines the organization of this structure. Operating within an environment configured in this way, SEAWAY would be able to send and receive the request and forecast objects via the main WAg. The main WAg would then create the

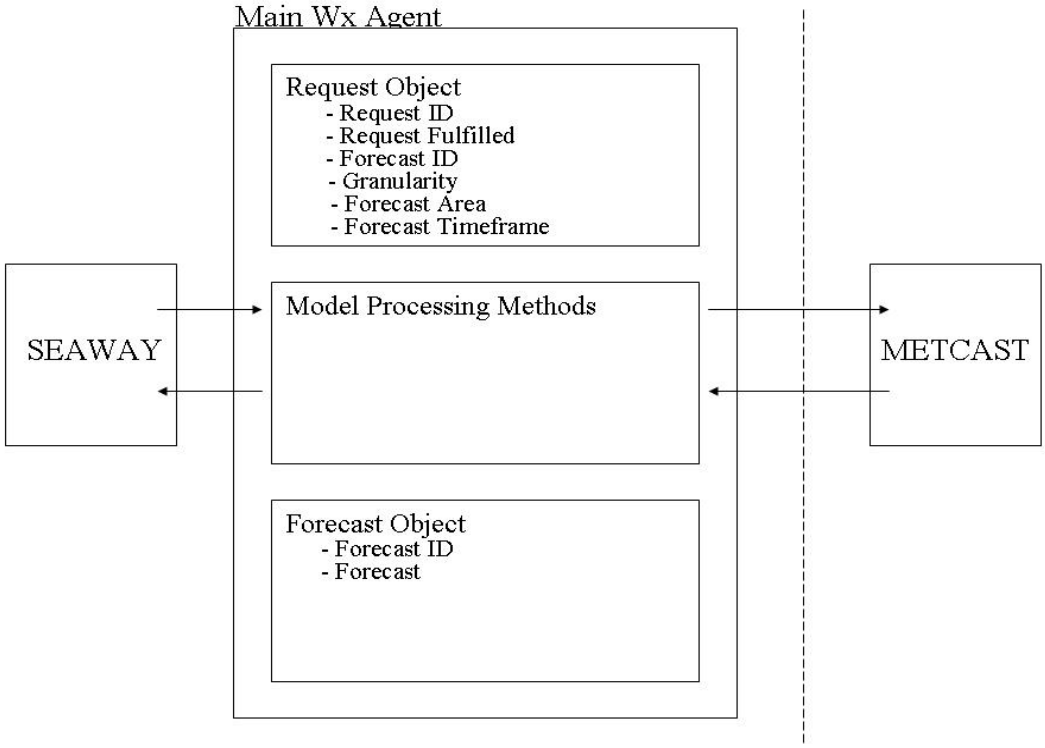

Figure 4. The WAg During Thesis Research.

desired request in MBL, send it to the Metcast server via an http connection, and download the desired model output. That model output could then be processed locally and used to create the final forecast object. The main advantage of this was added simplicity for the programmer. It was also apparent that since both types of objects were still created, http connections were used to send and receive information, and the model output was still processed in the same manner, the use of this design structure from a proof-of-concept approach was still valid. The main disadvantage to this sort of structure

was that if it were implemented Navy-wide, huge increases in bandwidth usage would result. In this case instead of sending small request or forecast objects, full model runs would be transmitted. In this type of scenario, similar requests could be made by different pieces of software and there would be no way of recognizing this and preventing the duplication of effort. In an academic environment, however, these effects were noted in case of a larger implementation, but deemed insignificant to the outcome of the research. It is within this structure of both the environment and the WAg that the testing presented in the following chapter was conducted.

V. TESTING

As we move into the testing phase of this thesis, it is important to fully describe where the WAg stands after the initial development stage. When activated, the agent uses a hardcoded request object, establishes an http connection with the FNMOC Metcast server, and sends a request for gridded COAMPS output, either wind or 12-hour precipitation, or WAM significant wave heights. The agent then establishes an input stream to read that model output into a data structure defined by the program. That data structure is then decoded so that three main arrays can be produced. Arrays are created for the latitude positions, the longitude positions, and the requested data. These arrays are then combined and passed into the main body of the forecast object. With a general understanding of the functionality the agent provides, we can now move forward and examine how to test it.

The testing of the WAg poses a unique challenge due the very nature of the software itself. This agent was designed to be a fresh, transformational look at how to handle meteorological and oceanographic data and best make it available to operational users. A major goal of the agent is to be able to bridge proprietary, insular applications (technological stovepipes) and allow a freer exchange of information within a FORCEnet environment. Due to its nature, therefore, when the WAg is operating perfectly, its presence would be only minimally apparent to the user. The only indication the user would have that the agent was functioning properly would be a visible integration of weather data into the planning process. In this sense, then, the WAg is either functioning properly as an entity that can obtain and process data or it is not. The idea of testing becomes a binary value where either the agent works completely or it does not. For this reason, we must examine other means of evaluating the agent and discuss their validity.

When considering how best to evaluate the effectiveness of the developed WAg, it may be important from one perspective to validate the forecast object output for correctness. This sort of testing mechanism would focus on evaluating the accuracy of the output data by comparing it with some sort of control. For example, the testing of this thesis could then consist of a side-by-side comparison of agent output with Joint METOC

Viewer (JMV) output. Even a qualitative comparison of these outputs would give us a feel for how accurately the WAg was handling the COAMPS and WAM grids and would ensure that data was not getting corrupted or mishandled during the packaging of the forecast object. It must be recognized, however, that this sort of testing would be largely trivial. Since both JMV and the WAg use the same source for their data, showing that the data remained unaltered within the agent program would not be a significant or rigorous evaluation. It is for this reason, that testing of the WAg should be more effect driven, and focus on how a properly functioning agent would contribute to warfare planning.

In a perfect testing environment, the effectiveness of the WAg would be evaluated as it imported meteorological data directly into the SEAWAY program. Unfortunately, this sort of integration would obviously require the full incorporation of the WAg by software developers from CDM Technologies, Inc into their program. Obviously, this was not always a practical expectation during the research and development of this thesis. As such, a scaled down examination of the effect or impact of this functionality must be evaluated instead. This can be accomplished by framing a warfare scenario within SEAWAY, and then exploring how importing wind, wave or precipitation data directly into this program would impact operational planning. This testing mechanism was deemed to provide a more significant evaluation of the value of the WAg, than a mere qualitative comparison.

A. THE SCENARIO

According to its basic training manual (CDM, 2003), SEAWAY was designed to support Marine Air/Ground Task Force (MAGTF) staff planning and was created for, and funded by, the Marine Corps Systems Command (MARCORSYSCOM) and the Marine Corps Warfighting Lab in Quantico Virginia. For this reason, it seemed most logical to create an amphibious assault scenario utilizing these kinds of forces to assess the agent impact. It also seemed logical to choose a geographic region that had current operational significance. As such, the Korean Peninsula was chosen as the target for this hypothetical assault. This is pictured in Figure 5. It seems reasonable to assume that at some point, if tensions concerning the development of nuclear capabilities by the North Korean government continue to escalate, an amphibious operation mounted from the Sea of Japan might be an all too real possibility. From this zoomed out view of the area of

interest we see some of the basic functionality of SEAWAY. The program provides a general outline of the geographic region, as well as several menu bars. Without going into a detailed description of each one, we can generally see that there is a vertical menu bar along the left side of the screen that represents the SEAWAY agents and there is a horizontal menu bar across the top of the screen for map navigation within the program. Here we can also see the aforementioned weather agent that is already a part of SEAWAY, however, the testing being documented here will highlight the added utility

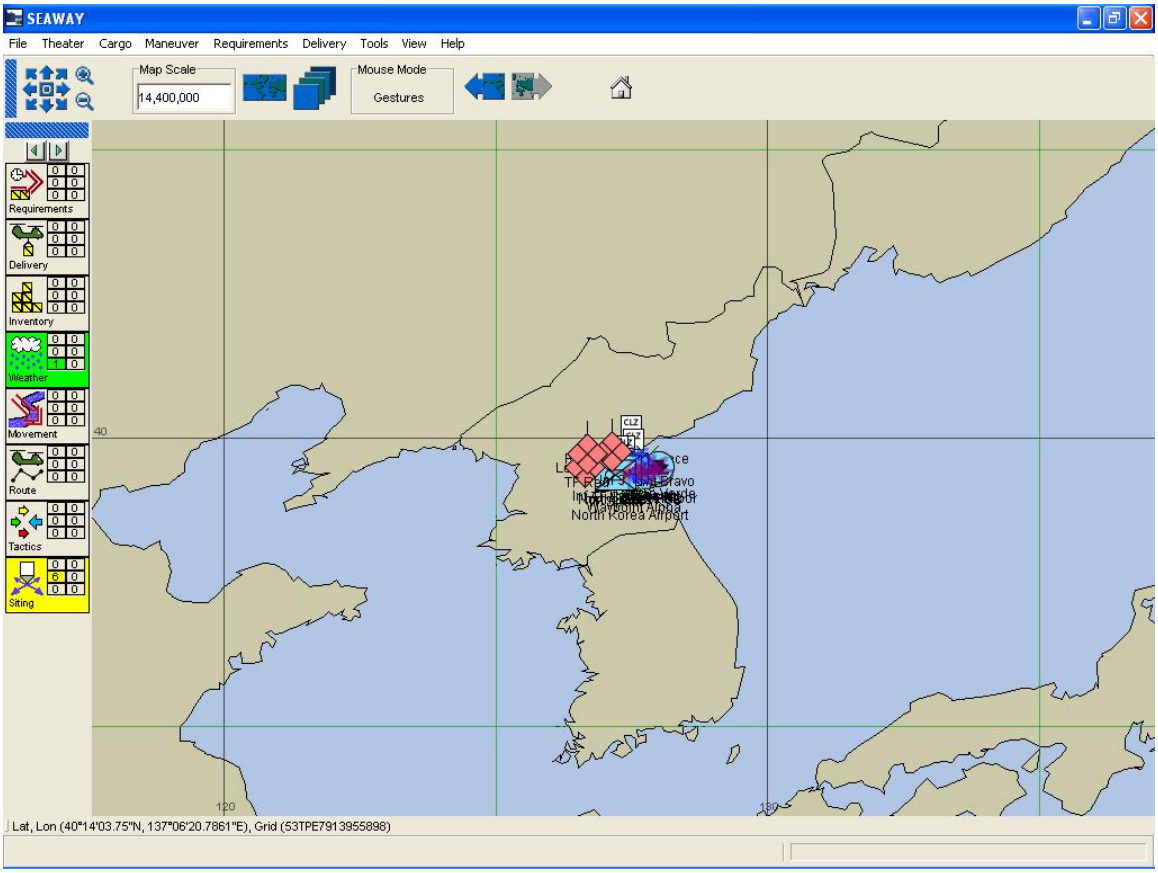

Figure 5. Zoomed-Out Scenario View.

that the WAg will provide. The units created for this supposed operation are also visible, and will be discussed next utilizing a zoomed in view of the operational area of interest. The top menu bar also shows the current map scale of 1:14,400,000.

As we zoom in closer over the west coast of North Korea, the particulars of the scenario become clearer (Figure 6). We see a Sea Base established offshore consisting of the Essex Amphibious Readiness Group (ARG) with three ships, the USS ESSEX (LHD

2), the USS MESA VERDE (LPD 17), and the USS PEARL HARBOR (LSD 49). A Marine tactical base has been established ashore, which is designated as North Korea FOB. There are also three Marine Task Forces (TF) securing the area, TF Axe, TF Sword, and TF Spear, each of which has a weapons company and/or an infantry company. There are two craft landing zones, CLZ Eagle and CLZ Pelican, and one Helicopter Landing Zone, HLZ Vulture. Delivery routes for logistical support are represented between the ESSEX ARG and all four landing areas. The routes highlighted in blue are for utilization by landing craft only, the routes highlighted in red are for use

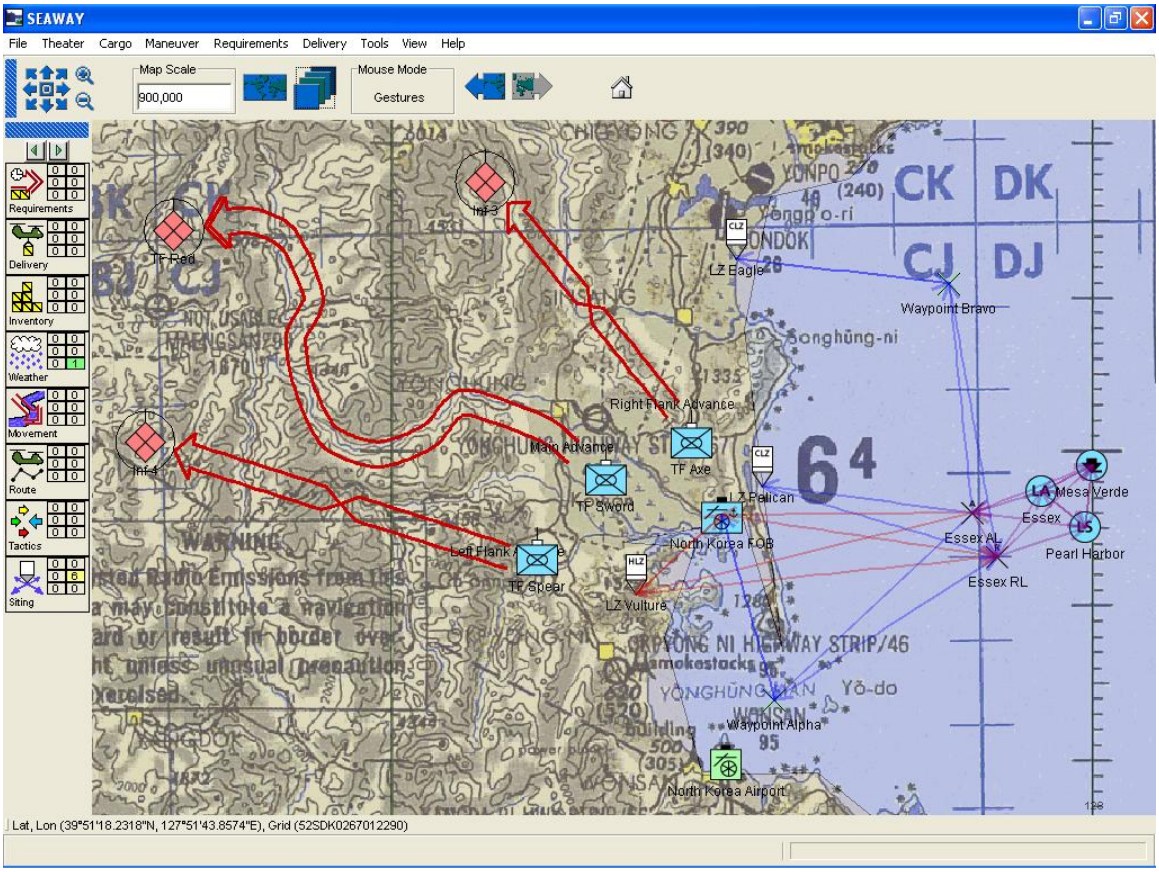

Figure 6. Zoomed-In Scenario View.

by aircraft only, and the routes highlighted in purple are for use by both landing craft and aircraft. There are three enemy units inland that are the objective of the assault, TF Red and supporting infantry units, Inf 3 and Inf 4. The companies from TF Sword will conduct the main ground assault of this operation on TF Red, with supporting helicopter assaults on the two enemy infantry units from TF Axe and Spear. With a basic warfare

scenario in place we can now examine how weather parameters may impact the planning of this amphibious assault and, by extension, the importance of the WAg.

This scenario was evolved to this point temporally because it highlights the functionality of SEAWAY. It would have been easy to create an offshore Sea Base and depict an initial assault into the landing area. With ground troop ashore executing various schemes of maneuver, however, the complexity of SEAWAY is more effectively showcased. Within this sort of scenario, SEAWAY can calculate usage rate for supplies, determine the resulting logistical requirements, and evaluate the various delivery options. This scenario could have been limited to an initial amphibious assault, but it was determined that a scenario containing a full description of established bases and advancing units would be more effective.

B. THE WEATHER IMPACT

It should be fairly obvious, to even the most casual observer, that within this operational scenario the proper assessment of environmental conditions would play a vital role in the success of the assault. In order to evaluate this impact, we will input in turn each of the three types of data that the WAg is capable of providing. In some instances, we will see that the delivery capabilities of the agent coincide nicely with the display capabilities of the SEAWAY program. In other cases, the WAg will provide information that would seem desirable to warfare planners, but that is not currently supported by the latest version of SEAWAY. This process is not intended to imply that there are any gaps in the utility provided by CDM's software, but rather to help define some of the future functionality that should be required by the METOC community within FORCEnet. The reader should understand that SEAWAY was produced to be a logistical decision-aid, not an environmental assessment tool. Studies such as this one are required to develop fully the required capabilities of the weather agent portion.

1. 12-Hour Precipitation

The amount of precipitation that falls over a combat area can have a significant impact upon the effectiveness of employment of equipment and personnel. For this reason, it seemed important to introduce an area of rainfall into the assault scenario and observe how it affects the various assets represented within SEAWAY. COAMPS calculates this parameter and Metcast returns amounts of precipitation in kilograms per

square meter per latitude/longitude coordinate. This correlates well with SEAWAYs display capabilities. Within SEAWAY the user can define weather areas that are bounded polygons representing environmental conditions. These environmental conditions can include dust storms, fog, hail, tropical storms, hurricane, and precipitation varying in intensity from drizzle to thunderstorms. When returning a forecast object that contains 12-hour precipitation data, the magnitude at each grid point could be correlated to the intensity of the precipitation. This would allow the creation of an appropriately labeled bounded polygon within SEAWAY that was denoted by the proper weather symbology.

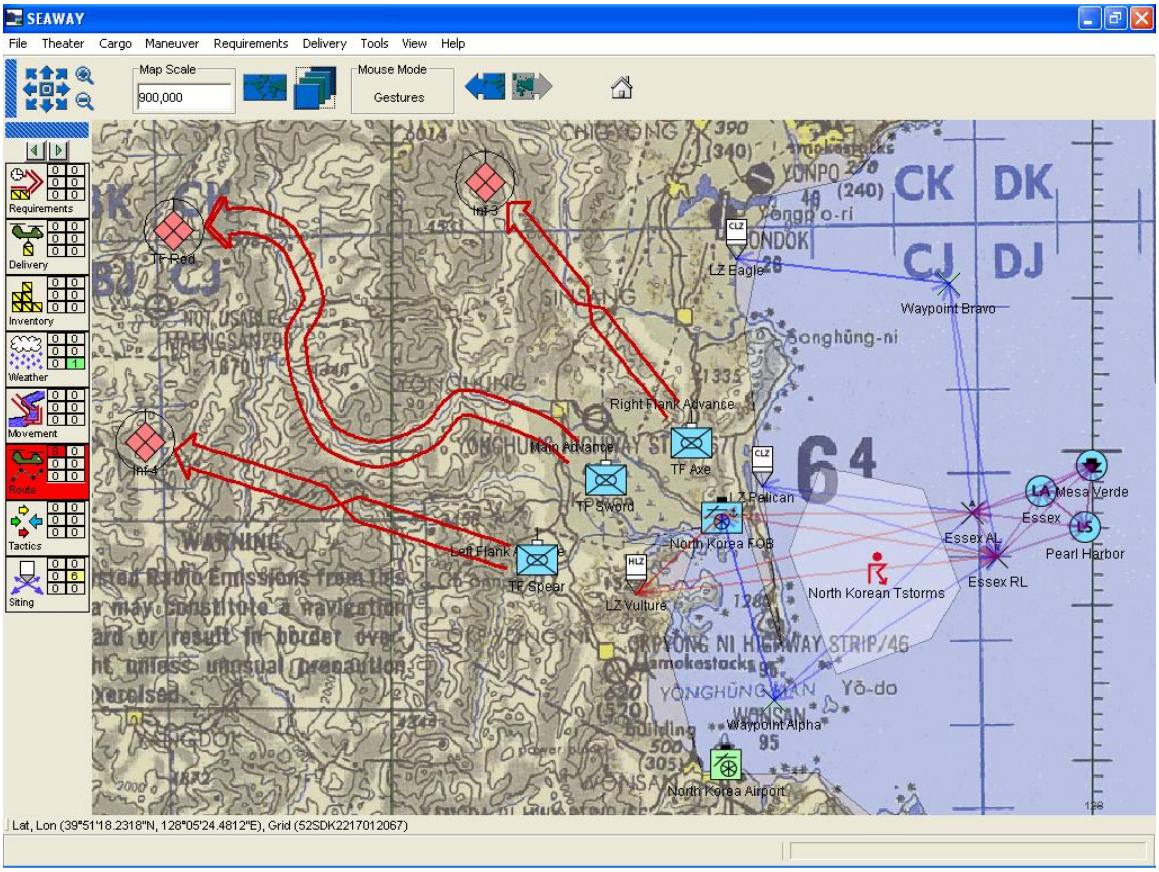

Figure 7. Area of Thunderstorms Off of the North Korean Coast.

We examine the impact this kind of data would have on operational planning within SEAWAY by placing an area of thunderstorms and rain between the ESSEX ARG approach and retirement lanes and the North Korean shore. In Figure 7 we can see that this area is labeled North Korean Tstorms, and that it has an almost immediate effect. Our first indication that there is a problem with our planned operations is that the Route agent

on the left menu bar turns red and displays the number 8. This indicates that there are unfavorable conditions present that will severely degrade or prevent the movement of supplies along the assigned delivery routes. Obviously, the air and sea routes that connect the ESSEX ARG and CLZ Pelican, HLZ Vulture, and the North Korean FOB will all be severely impacted by these conditions. By clicking on the Route agent, we get the more detailed description of the individual alerts as shown in Figure 8. Here, a text message gives a more detailed description of the problems encountered along each of the eight routes that run through this particular hazardous area. In each case the message ultimately indicates, "No transport operations on this route possible."

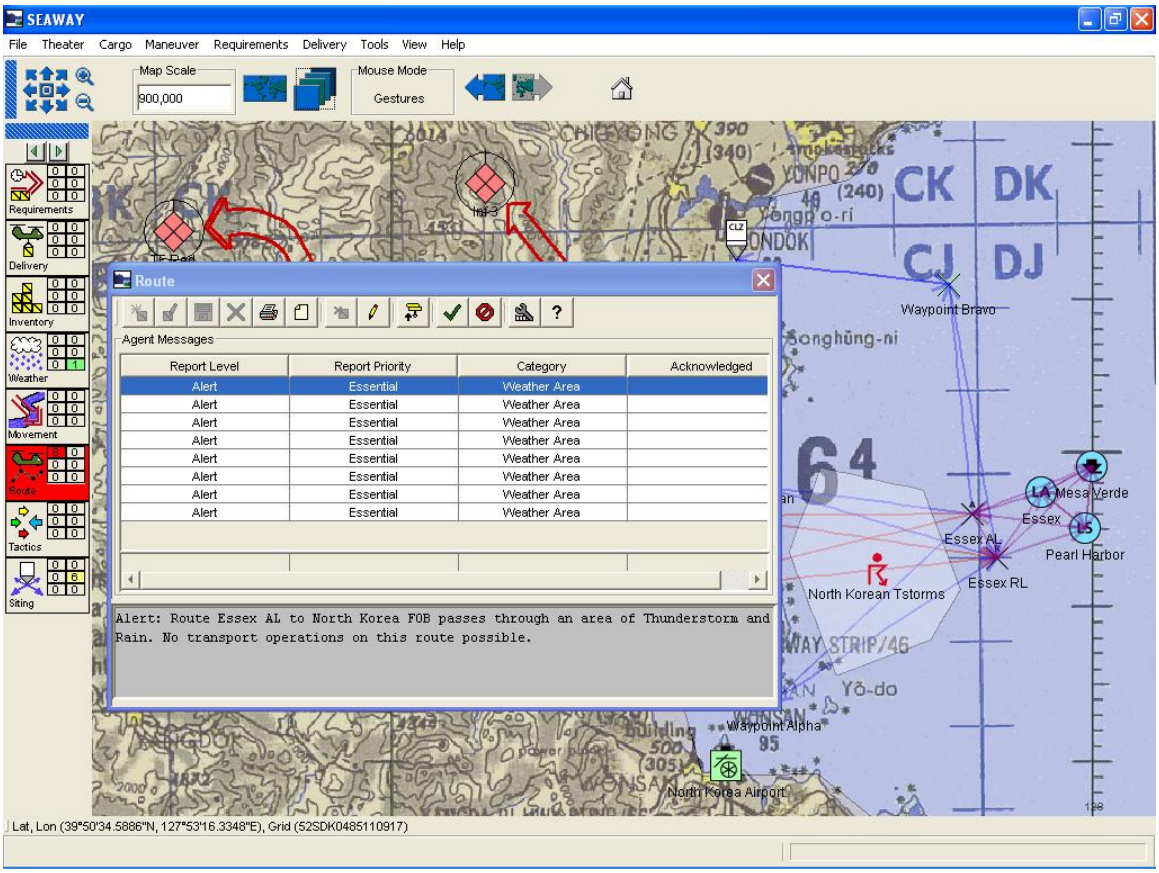

Figure 8. Alert Messages in Response to Offshore Thunderstorms.

For this particular forecast parameter, we see that the SEAWAY Weather and Route agents work together very effectively to compare forecast conditions with asset capability. This intelligent comparison is a desirable capability that could be an effective example of using of available data when needed. Providing reach-back capability to

provide this data as required is a solid example of FORCEnet implementation, and is the type of utility the WAg will provide when incorporated into the SEAWAY program. To develop this idea further, ultimately we should be working towards an environment where all available data is integrated and available at the push of a virtual button. This data could include Doppler radar imagery from near-by land stations, in-situ measurements from advance Special Forces units, available satellite imagery of the area of interest, and even tactical radar assessments of the environment from offshore naval assets. This particular weather parameter is also a good example of an environmental condition that SEAWAY does a good job of representing. The program is able to qualitatively break out areas of precipitation based on intensity and then visually represents them to the user in a clear, easy to understand manner. The warfare commander gets immediate feedback on how this type of weather will impact the employment of their assets within the same software package that he or she is using to plan the operation. The technological stovepipe is effectively crossed, or broken-down, in a FORCEnet-like fashion.

2. Significant Wave Height

Significant wave height is another meteorological parameter that can play a major role in planning and executing amphibious operations. As the term “amphibious” implies, many of the delivery vehicles for equipment and personnel travel either on or through the water. Thus, the height of the waves along the surface of the water can often determine whether or not an assault can be executed or supported. This is represented well by significant wave height, which is defined as the average height of the highest one-third of the waves present. WAM calculates this parameter and Metcast returns it in terms of height in meters per latitude/longitude coordinate. The most noticeable difference in how this parameter is represented within SEAWAY is that the user cannot define a weather area to display this data. Instead, the user enters the significant wave height as a single value within an overall weather report. Immediately, the defining of this parameter in such a manner should be cause for further examination.

When input into the SEAWAY program, significant wave height is represented in a much more simplistic form than 12-hour precipitation. As stated, this parameter is added as a single entry in the weather report section. Figure 9 shows, however, that the

SEAWAY agents still view this piece of data as significant to operations. In this test the user inputs a significant wave height of 10 feet. Unlike before, this time the Route agent did not register an alert. Instead, the SEAWAY Weather agent returned an alert stating that under these conditions, “No landing craft operations possible.” There was also no

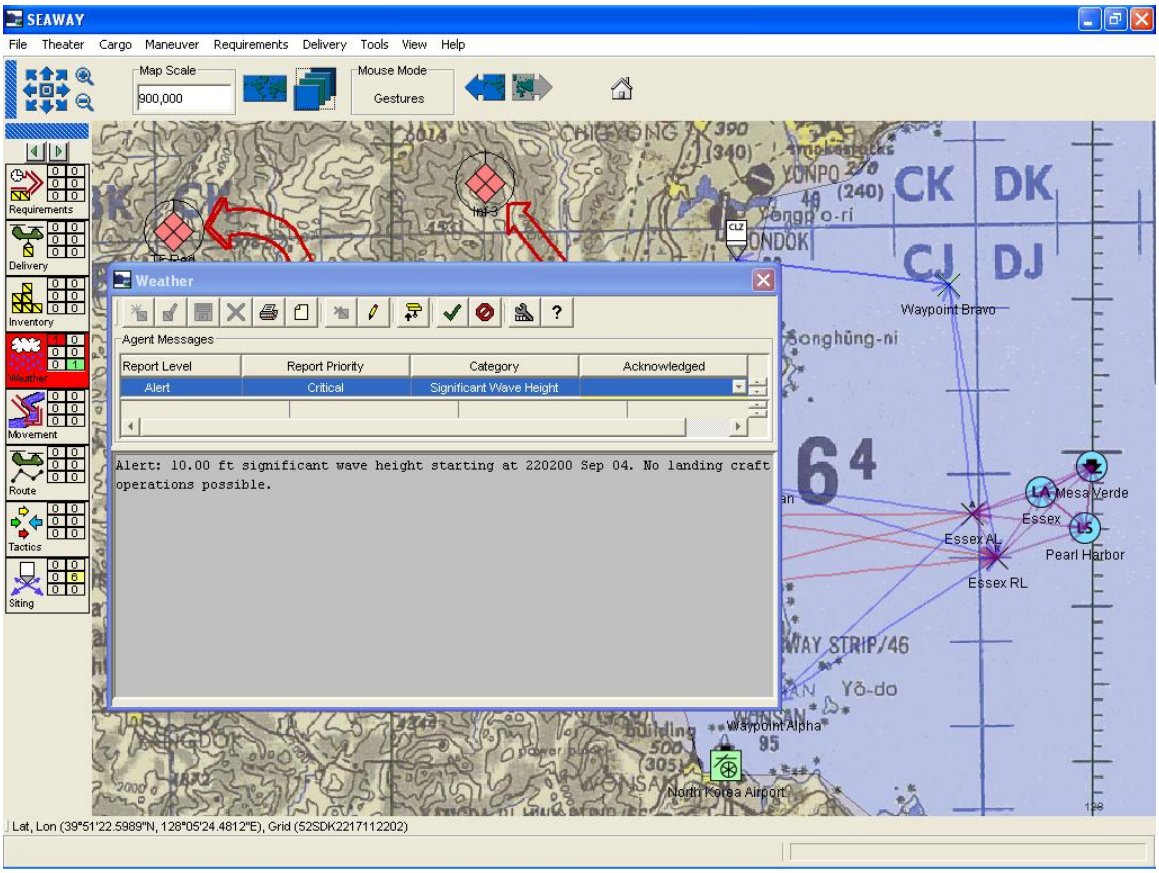

Figure 9. Alert Messages in Response to a Significant Wave Height of 10 Ft.

visual representation of this prohibiting parameter within the mapped environment since it was not associated with a coordinate location. This result is significant since it is readily apparent that a single value is not sufficient to describe significant wave height within the natural environment. This is a complex piece of oceanographic data that can vary significantly, especially in a littoral environment. The WAg returns significant wave height in an array of latitudes and longitudes with corresponding heights in meters. This could be modified to only return the highest value in the array and convert it to feet in support of SEAWAY, but doing so would be at the expense of a more accurate depiction of the ocean surface.

It should be noted that a savvy user of SEAWAY could most likely work around the less detailed depiction of this parameter. For example, if an operator was interested in a particular sea delivery route they could enter the significant wave height along that particular path and then observe the resulting assessment. A similar change could be made for each route of interest in turn. Of course, this sort of manipulation by the user would be somewhat counterproductive to a full and descriptive representation of the operational environment and its impact on the movement of personnel and supplies.

3. Surface Wind

In addition to 12-hour precipitation and significant wave height, surface wind is also an environmental parameter that can have a major impact on most military operations. In the case of an amphibious assault, surface winds can play a major role in troop movement, support, and Sea Basing. COAMPS calculates this parameter and Metcast returns it in terms of the u and v components in meters per second per latitude/longitude coordinate. As with significant wave height, SEAWAY does not provide functionality to display this as a weather area. Instead, as before, surface wind is entered as a single value for both wind speed and direction within the operational area. We will see that this characterization of surface wind, and how the SEAWAY program incorporates this parameter, is worth some discussion.

When input into SEAWAY, wind speed and direction has little or no effect on the planned operations. In Figure 10, a wind speed of 50 nautical miles per hour (knots) from a direction of 135 was entered into the program. This entry resulted in no apparent effect within the warfare-planning environment. None of the system agents registered any kind of alert notification, and there was no visual depiction of this wind speed or direction displayed to the user. There was no noticeable change made within the planning environment from the viewpoint of the operator. This observation was notable for several reasons. First, from an operational standpoint, there is no doubt that 50-knot winds would severely hamper the movement of troops ashore as well as the stability of the established Sea Base. Second, the depiction of wind as a singular value is a poor representation of this parameter as it exists within the natural environment. As with significant wave height, wind is an environmental parameter that can differ substantially over short distances. This is especially true within the littoral environment where topography can

significantly change wind direction and friction can reduce wind speeds. Third, it should also be noted that the variation of this parameter in the vertical is not addressed by SEAWAY. As previously stated, this parameter is listed within the SEAWAY weather report as simply “wind speed” and “wind direction”. It is not even explicitly stated whether this particular value is surface wind or wind at some particular atmospheric level, as it would pertain to aviation operations. Within this program, however, this distinction may not be a significant one, since regardless of the input value there is no noticeable resulting effect. In designing the WAg, it was stated that this wind parameter would be defined as surface wind, which would be further defined as 10-meter winds. As

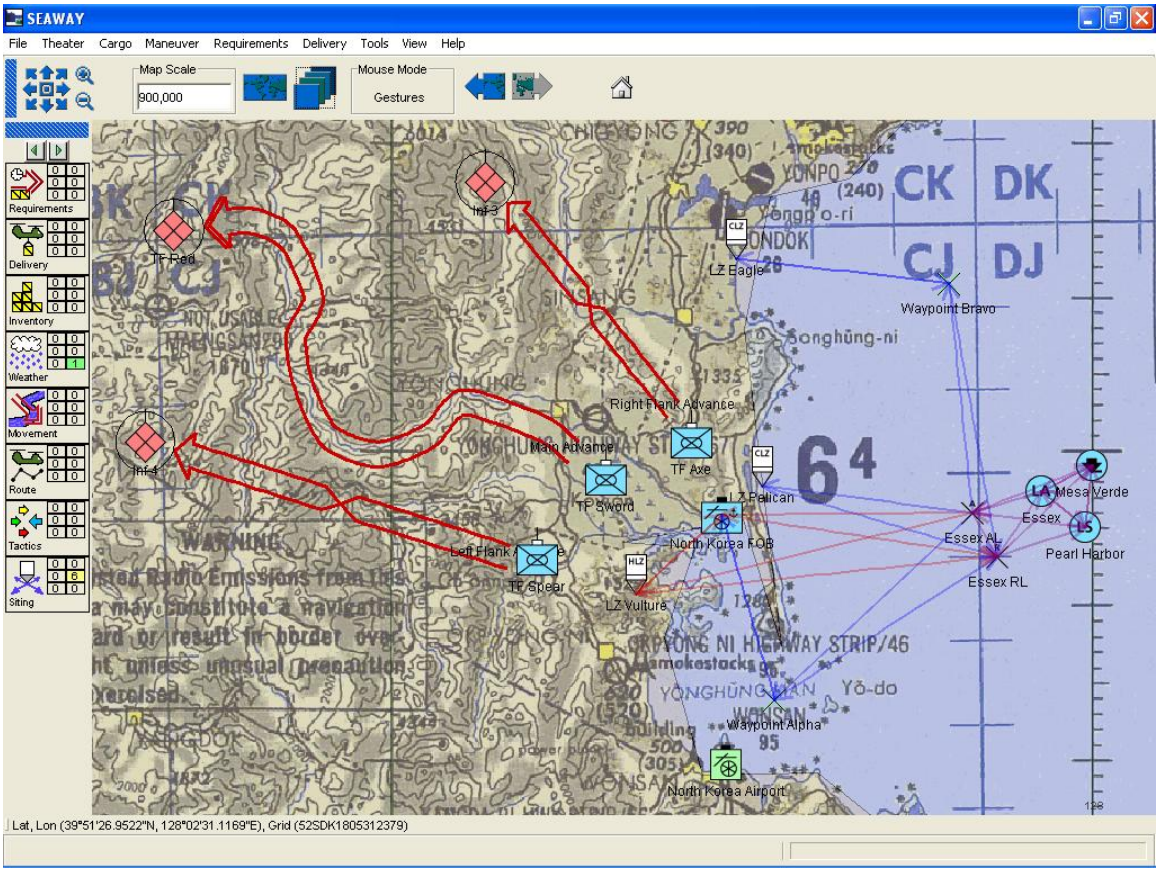

Figure 10. The SEAWAY Environment with a Southeast Wind of 50 knots.

such, the WAg was designed to return an array of latitude and longitude coordinates with corresponding wind speeds in meters per second and wind direction in degrees. As before, this output could be changed to produce singular values, but at the expense of a more accurate depiction of the surface wind environment.

C. APPLYING THIS TESTING TO FORCENET

This sort of testing allows us to form a vision of the functionality and utility that would be available to the user in a perfect FORCEnet environment. In a system where information is freely exchanged and transmitted data could be easily shared, added complexity could be introduced into the display capabilities. Higher resolution gridded output for every environmental parameter could be used instead of the singular values currently considered for some forecast items. Additional parameters, such as directional wave spectrum, could also be incorporated into the WAgs delivery capabilities. Within this kind of system, sensor information from the units within the Sea Base could also be used for through-the-sensor assessments of the environmental conditions. Users would then be able to access in-situ data via the program GUI. Temporal changes could also be expressed, which would enable warfare commanders to focus on the windows of optimal conditions instead of evaluating point forecasts for specific time periods. The forecast data could also be contained within more complex data structures, such as a representation of winds that included vertical distribution along with the horizontal. The application of the ideas presented via this thesis within the context of a FORCEnet environment open the door to the adequate delivery of virtually any environmental parameter.

It should be noted here that even upon the completion of this thesis, research into the continued development, refinement, and implementation of this agent will continue. The author will be writing an additional thesis over the next six months that will focus more on the computer science aspects of this project. The planned improvements will include getting the WAg fully integrated into SEAWAY, conducting more refined impact assessments, and expanding the utility provided by the WAg. The WAg will also be run on a remote data server with the local version of the agent running alongside the command-and-control tool, so that request and forecast objects can be exchanged. This thesis served as a proof-of-concept consideration of how agents can be implemented within the METOC community, the follow-on thesis will take further steps to make that implementation a reality.

The observations noted within this chapter are not presented as a criticism of the SEAWAY program. As examples to the contrary, the way in which SEAWAY represents

precipitation and incorporates vehicle-operating parameters into the assessments of its agents can be seen as excellent ideas. Rather, these observations are intended as examples of how this sort of testing can help define future meteorological or oceanographic requirements for FORCEnet. Funding is currently being allocated and distributed for the production of systems that can provide the functionality that would be required in support of FORCEnet. This testing emphasizes that in order for the METOC community to ensure that this software will be able to effectively handle and accurately represent environmental parameters, we must provide developers with a set of standard functional requirements. Failure to do so may result in the production of software that displays this information in a format that is either undesirable or ineffective in describing the environment.

Some of these functional requirements should establish the kinds of visual representation our community desires for weather parameters. This type of statement could include, for example, an assertion that all environmental data will be displayed as polygons defined by latitude/longitude boundaries referenced to one particular datum. It could also require that it be possible to overlay these polygons on whatever mapping or display capability the software would incorporate. Another requirement could be whether or not it was necessary to include the representation of standard weather symbology within these environments. By detailing exactly the display functionality that was desired, or required, we could avoid ineffective visual representation of environmental parameters within FORCEnet programs.

Other functional requirements set forth by the METOC community could establish how the exchange of weather data would occur. For example, in a net-centric, FORCEnet environment the http standard could be adopted. Drilling down further, how each type of parameter would be passed could be detailed. For example, within this thesis it was defined that the 12-hour precipitation data would be passed as an array of values. Within this array, latitude and longitude would be represented by parsed integers and precipitation by integers. By defining how the data our community provides should be represented and exchanged within a FORCEnet environment, we take the necessary steps required to ensure that the proper interfaces will be in place as needed.

Defining functional requirements can be a powerful integration tool. These ensure that the groundwork is in place to freely exchange environmental data, and that the software architecture present can accommodate meteorological and oceanographic parameters. With this functionality available, the METOC community would be poised to ensure that the consideration of environmental conditions would be fully incorporated into the FORCEnet structure and, thus, into the regular planning and execution of all operations.

VI. CONCLUSIONS

There are several main conclusions that can be drawn from this thesis. The first of these is that software agents are a viable means of processing data within the METOC community and, in larger terms, within the Department of Defense. The second conclusion is that this agent architecture is an implementation that is in direct support of FORCEnet. The final main conclusion of this thesis is that the creation of defined sets of METOC requirements are vital to ensure that weather parameters are effectively and fully integrated into the developing FORCEnet environment.

A. METOC TRANSFORMATION THROUGH APPLYING AGENTS

As described in Chapter II and illustrated in Chapter V, software agents are a versatile and powerful means of processing and manipulating data. Within the context of this thesis, the WAg was used to download numeric model output, process that raw data, and then package it in a usable format. This was a relatively simple example of a reactive agent, but this sort of implementation serves as a proof-of-concept starting point for more advanced applications. In future versions, the WAg might employ more cognitive processing approaches. An example of this would be an agent that could track forecasts and verifications for different models in certain geographic areas of interest. This agent could then determine which model was exhibiting better performance in each region and create an ensemble forecast for its customer. Cognitive agents might also be able to assess mission parameters and return windows of time during which optimal conditions exist. Two other students at NPS are currently researching thesis topic related to agents with this type of complexity. One is looking into the use of agents to produce composite hurricane track forecasts using genetic scoring algorithms to assess various model performances. The other is researching the use of agents to automatically consider vertical wind shear in the planning and execution of ordinance delivery. This type of powerful processing could transcend the traditional method of point forecasting and shift focus to achieving the ultimate goals of the warfighter instead. Software agents that evaluate environmental conditions in different ways could replace TDAs as a means of

providing tailored support on a 24/7 basis. The power of this technology is derived from the realization that there is virtually no limit to the complexity with which agents can be imbued.

The implementation of autonomy within an agent environment could also change how manpower is structured within the METOC community. In a time if fiscal constraint, more basic forecasting tasks could be performed by complex agents that could create, assess, and revise products as necessary. With less routine duties to complete, additional personnel might be free for more effective employment. Better utilizing our manpower resources would streamline budget requirements, and might possibly ease our transition to a more compact fighting force. This additional autonomy would be the next iteration of the WAg development, which would move beyond the reactive agent presented here.

B. SUPPORTING FORCENET THROUGH AGENTS

For several reasons, the use of software agents is a concrete example of the implementation of main FORCEnet tenets. This thesis illustrated a case where this technology was used to cross the boundaries of proprietary, insular software. Crossing these pre-existing boundaries allows information to be passed between otherwise incompatible software. This freer interchange of data and information is in direct support of the FORCEnet concept, where every data repository becomes a node in the larger net-centric environment. Each node, then, becomes capable of delivering data on an as-needed basis. Moreover, the delivered parameters arrive in a format that is immediately usable to the native, requesting software. Agents also support the focus on reach-back capability via web services. The WAg utilized http connections to send requests and receive data. This enables a globally distributed usage that corresponds directly with the actual worldwide commitments of the U.S. Armed Forces. By using common standards to create this software and ensuring that implementations are platform and operating system independent, extensibility and forward/backward compliance is secured. For all of these reasons, software agents are a suitable implementation of FORCEnet tenets.

This sort of structure would also be conducive to implementing future advances in weather sensing or forecasting technology. We may not be far from the day when individual ground troops carry weather sensors on their gear harnesses for in-situ assessments of the environment as they advance. Combined with rapid microscale

modeling driven autonomously by agents, this would allow immediate impact assessments and revisions of the operational picture. As through-the sensor technology continues to be developed, agents could easily be modified to understand and process that data in new and innovative way. Once the groundwork is in place for the agent-based processing and delivery of data, this architecture will be uniquely sited for rapid adaptation to accommodate new sources of information.

C. DEFINING COMMUNITY REQUIRMENTS

In a succinct way, the testing of this thesis served as an example of the need for the METOC community to fully define and promulgate its functional requirements for developing FORCEnet software. The easiest way to ensure that computer programs will be able to share information is to establish interfaces that are built into each system. It can be assumed that money is being allocated in order to produce FORCEnet compliant computing systems. The sooner the METOC community reaches a consensus on the data we wish to represent, how that data will be structured, and the types of display capabilities required, the easier it will be to incorporate it into developing technologies. This thesis defined three of these parameters: surface wind, 12-hour precipitation rates, and significant wave height. This example could be extended to include the full range of desired meteorological and oceanographic parameters.

Instead of simply defining system requirements, our community could also develop pre-packaged software entities, such as the WAg, for developers to incorporate. As a community if we expanded agent development and then handed it to developers as a unit we would ensure the capabilities we desired for use were incorporated into their systems. At the very least, creating this sort of software bundle would allow us to define the API we wished to work through.

D. TRANSFORMATION: AN OFFER WE CAN'T REFUSE

There can be little doubt that technology will continue to advance at its current exponential rate, whether we like it or not. As a result, we must either move with it or be surpassed by it. The transformation and advancement of the METOC community must incorporate the new and emerging Information Age technologies becoming available virtually every day. It is through the effective implementation of these tools that we will maintain the competitive edge in warfighting, and more specifically the assessment of the

natural environment, which our country has enjoyed for so long. Re-evaluating how we transmit data and incorporate it into our planning processes is a healthy exercise that should be repeated frequently in order to ensure it is still relevant and current. The utilization of agent-based software as a means of processing data and delivering information should be a part of this kind of periodic evaluation. For all of the reasons presented here, software agents are a versatile, powerful means of computing that will become more relevant as autonomy and cognitive decision making begins to permeate the world of computing.

LIST OF REFERENCES

Arthur, W. B., 1994: Inductive Reasoning and Bounded Rationality. [Available online at http://www.santafe.edu/arthur/Papers/El_Farol.html]

CDM Technologies, Inc., 2003: *SEAWAY Version 2.0 Basic Training Manual*. [Available from 2975 McMillan Ave. Suite 272 San Luis Obispo, CA 93401]

Cebrowski, A. K.: The Office of Force Transformation: What is Transformation?. [Available online at http://www.oft.osd.mil/what_is_transformation.cfm]

Gentges, D., Java programmed computer code. [Available from Fleet Numerical Meteorology and Oceanography Center 7 Grace Hopper Ave. Monterey, CA 93943]

Gunderson, C. R. & Higgins, S., 24 February 2004: Presentation to OC4270 (Tactical Oceanography) at the Naval Postgraduate School: Net-centric Operations.

http://geoengine.nga.mil/muse-cgi-bin/rast_roam.cgi. Last accessed August 2004.

http://www.metnet.navy.mil/Metcast/. Last accessed August 2004.

http://www.metnet.navy.mil/Metcast/Code/grid-serve.scm. Last accessed August 2004.

https://www.fnmoc.navy.mil/PUBLIC/MODEL_REPORTS/MODEL_SPEC/coamps2.0.html. Last accessed August 2004.

Pohl, J., cited 2004: Transitioning from Data to Information. [Available online at http://www.cadrc.calpoly.edu/pdf/data_information.pdf.]

Reynolds, C. W., 1987: Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21, 25-34.

Schach, S. R., 2002: *Object-Oriented and Classical Software Engineering*. 5th ed. McGraw-Hill, 290-318.

Weiss, G., Ed, 1999:*Multi Agent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 1-77.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Carlyle H. Wash
   Naval Postgraduate School
   Monterey, California

4. Dr. Neil C. Rowe
   Naval Postgraduate School
   Monterey, California

5. Dr. Philip A. Durkee
   Naval Postgraduate School
   Monterey, California

6. LCDR(sel) Jonathan J. Vorrath
   Naval Postgraduate School
   Monterey, California